IN THE CLAIMS

Please amend the claims as follows.

1. (Withdrawn) A method of generating code for scheduling the execution of binary code translated from a source format to a target format, wherein the source format differs from the target format, said method comprising the steps of:

(a) identifying a set of target instructions semantically equivalent to a given source instruction, and wherein the set of target instructions is identified in a translation template associated with a given source instruction, said template being a component of a translator program for translating instructions in the source format into instructions in the target format;

(b) identifying data dependencies in said target instructions by analyzing the set of target instructions; and

(c) assigning an identifier to one or more of said target instructions for use by a code analyser in scheduling the processing of said set of target instructions in accordance with the identified data dependencies, and wherein data dependencies are represented by a directed acyclic graph, and the identifying step identifies said dependency signaling an appropriate edge in the set of target instructions to said code analyser.

2. (Canceled)

3. (Withdrawn) A method according to claim 1 in which the analysis of the target instructions is carried out prior to the compilation of the translation templates into said translator program.

4. (Withdrawn) A method according to claim 1 in which the identifiers are assigned to said target instructions prior to said translator program being compiled.

5. (Withdrawn) A method according to claim **1** in which said code analyser optimizes the translated code for processing in a parallel processing environment by using the identifiers.

6. (Canceled)

7. (Withdrawn) A method according to claim 1 in which each translation template is associated with a corresponding analysis routine that generates said code for scheduling the execution of said translated code.

8. (Withdrawn) Apparatus for generating code for scheduling the execution of binary code translated from a source format to a target format, wherein the source format differs from the target format, wherein said apparatus comprises a processor and storage means, and said apparatus arranged to be responsive to

(a) a set of target instructions semantically equivalent to a given source instruction stored in storage means;

(b) a translation template for identifying the set of target instructions, the translation template being associated with a given source instruction, said template being a component of a translator program for translating instructions in the source format into instructions in the target format stored in storage means; and comprising:

(c) an instruction analyser for analysing the set of target instructions using the processor to identify data dependencies in said target instructions;

(d) a dependency identifier for assigning an identifier to one or more of said target instructions for use by a code analyser using the processor in scheduling the processing of said set of target instructions in accordance with the identified data dependencies; and

(e) a directed acylcic graph for representing data dependencies, and the identifier is arranged to identify said dependency signaling an appropriate edge in the set of target instructions to said code analyser.

9. (Canceled)

10. (Withdrawn) Apparatus according to claim 8 wherein the instruction analyzer is arranged for arranging the target instructions prior to compilation of the translation templates into said translator program.

11. (Withdrawn) Apparatus according to claim 8 wherein the dependency identifier is arranged to assign the identifier to said target instructions prior to compilation of said translator program.

12. (Withdrawn) Apparatus according to claim 8 further including said code analyzer, said code analyzer being arranged to be responsive to the identifiers for optimising the translated code for processing in a parallel processing environment.

13. (Canceled)

14. (Withdrawn) Apparatus according to claim 8 further including a plurality of said translation templates, in which each translation template is associated with a corresponding analysis routine for generating said code for scheduling the execution of said translated code.

15. (Withdrawn) A computer readable medium or storage device storing coded indicia for causing a data processor arrangement to perform the method of claim 1, when executed by a processor.

16. (Withdrawn) A binary code translator that operates between any processor or platform types for translating binary code from a source format to a target format for execution on a target processor, the source format differing from the target format, the translator comprising a computer readable medium or storing device storing coded indicia adapted to be read by a data processor arrangement, the coded indicia including:
(a) a set of translation templates, each template arranged for providing a set of target format instructions which together are semantically equivalent to an associated source format instruction, wherein each translation template is associated with a given source instruction, said template being a component of a translator program for translating binary code in the source format into the target format;

(b) a set of data transformation routines arranged to transform data from a source format instruction into the appropriate parts of each target format instruction provided by the corresponding translation template; and
(c) a set of analysis routines arranged to identify data dependencies in a template for causing generation of data for use by a code scheduler in scheduling the execution of translated code on said target processor, and wherein data dependencies are represented by a directed acyclic graph, and the identifying step identifies said dependency signaling an appropriate edge in the set of target instructions to said code analyser.

17. (Withdrawn) A binary code translator according to claim 16 arranged to operate dynamically at the run time of an application program being emulated.

18. (Withdrawn) The method of claim 1 wherein the code analyzer schedules the processing of said set of target instructions in accordance with the identified data dependencies.

19. (Withdrawn) The apparatus of claim 8 in combination with the code analyzer arranged to be responsive to the identifier assigned to one or more of the target instructions, the code analyzer being arranged for scheduling the processing of said set of target instructions in accordance with the identified data dependencies.

20. (Canceled)

21. (Previously Presented) A build and compilation process of a dynamic binary translator which operates between any processor or platform types and that translates a source code instruction into a target code instruction, wherein the source code differs from the target code, the process comprising:
(a) creating a repository of magic numbers, wherein each magic number is a unique integer representing one and only one variant of a source instruction so that the collection of all the magic numbers represents the set of all valid instructions possible of a source instruction set to

form a set of constants associating a unique string with a unique integer, wherein the string shortly describes a variant of the instruction which the magic number is supposed to represent;

(b) creating a repository of templates, wherein each template is a functionally equivalent sequence of target architecture instructions, for each source instruction, each template having temporary native registers for its operations and is compiled at the compile time of the binary dynamic translator as a function and whose name is derived from the corresponding magic number's associated string so that the templates are available as a routine/function at run-time;

(c) creating a table, indexed on magic numbers, containing the references to the corresponding templates so that at run-time the translator can pick up the appropriate template for a given instruction after ascertaining its magic number;

(d) creating a repository of template filler routines in which each template filler routine has the knowledge of both the source and target instructions; causing the template filler routine of a given template to extract dynamic components of the source instruction and fill the extracted items in an in-memory template to replace corresponding place-holders; causing the translator to call the template filler routine for a given instruction at run-time so a template filler routine repository is linked to corresponding templates and magic numbers;

(e) creating a repository of minimal decode/magic-number-routines; causing a minimal decode/magic-number-extractor routines repository to give a raw source instruction and return the magic number of that instruction;

preparing the dynamic binary translator to operate between any processor or platform types by compiling steps (a)-(e) together.

22. (Previously Presented) A process of operating the dynamic binary translator of claim 21 in response to a source instruction Si by performing a plurality of steps including:

(1) mapping Si to an appropriate minimal decode/magic-numbers-extractor routine to repository (MNEi);

(2) call the MNEi with Si as a parameter;

(3) causing MNEi to return the magic number mi;

(4) mapping mi to template ti;

(5) copying ti to a code generator buffer b;

(6) mapping mi to a template filler routine (TFRi);
(7) causing TFRi to extract dynamic compliments of Si and fill in the ti found at buffer b;
(8) going to S(i+1) if the current block is not yet finished;
(9) optimizing the sequence of instructions in buffer b for best tempera performance; and
(10) emitting optimized code from buffer b to target code cache Ci.

23. (Previously Presented) A method of operating a dynamic binary translator of claim 21 responsive to a source instruction Si in a source code so that the translator derives a target instruction in a target code that differs from the source code, the method comprising: using a repository of magic numbers, wherein each magic number is a unique integer representing one and only one variant of a source instruction so that the collection of all the magic numbers represents the set of all valid instructions possible of a source instruction set to form a set of constants associating a unique string with a unique integer, wherein the string shortly describes a variant of the instruction which the magic number is supposed to represent and performing a plurality of steps including:
(1) mapping Si to an appropriate minimal decode/magic-numbers-extractor routine to repository (MNEi);
(2) call the MNEi with Si as a parameter;
(3) causing MNEi to return the magic number mi;
(4) mapping mi to template ti;
(5) copying ti to a code generator buffer b;
(6) mapping mi to a template filler routine (TFRi);
(7) causing TFRi to extract dynamic compliments of Si and fill in the ti found at buffer b;
(8) going to S(i+1) if the current block is not yet finished;
(9) optimizing the sequence of instructions in buffer b for best tempera performance; and
(10) emitting optimized code from buffer b to target code cache Ci.

24. (Previously Presented) The build and compilation process according to claim 21, wherein the dynamic binary translator maps the source code instructions to the target code instructions using the templates.

25. (Previously Presented) The build and compilation process according to claim 21, wherein the dynamic binary translator outputs the translated target code instruction for processing on a target processor using the templates and fill and analysis routines during compilation of the dynamic binary translator.

26. (Previously Presented) A dynamic binary translator system for performing a build and compilation process of a dynamic binary translator which operates between any processor or platform types and that translates a source code instruction into a target code instruction, wherein the source code differs from the target code, the process comprising:

(a) creating a repository of magic numbers, wherein each magic number is a unique integer representing one and only one variant of a source instruction so that the collection of all the magic numbers represents the set of all valid instructions possible of a source instruction set to form a set of constants associating a unique string with a unique integer, wherein the string shortly describes a variant of the instruction which the magic number is supposed to represent;

(b) creating a repository of templates, wherein each template is a functionally equivalent sequence of target architecture instructions, for each source instruction, each template having temporary native registers for its operations and is compiled at the compile time of the binary dynamic translator as a function and whose name is derived from the corresponding magic number's associated string so that the templates are available as a routine/function at run-time;

(c) creating a table, indexed on magic numbers, containing the references to the corresponding templates so that at run-time the translator can pick up the appropriate template for a given instruction after ascertaining its magic number;

(d) creating a repository of template filler routines in which each template filler routine has the knowledge of both the source and target instructions; causing the template filler routine of a given template to extract dynamic components of the source instruction and fill the extracted items in an in-memory template to replace corresponding place-holders; causing the translator to call the template filler routine for a given instruction at run-time so a template filler routine repository is linked to corresponding templates and magic numbers;

(e) creating a repository of minimal decode/magic-number-routines; causing a minimal decode/magic-number-extractor routines repository to give a raw source instruction and return the magic number of that instruction;
preparing the dynamic binary translator to operate between any processor or platform types by compiling steps (a)-(e) together.

27. (Previously Presented) The dynamic binary translator system according to claim 26, wherein the dynamic binary translator maps the source code instructions to the target code instructions using the templates.

28. (Previously Presented) The dynamic binary translator system according to claim 26, wherein a fill and analysis routine generator arranged to be responsive to the templates for generating fill and analysis routines for identifying the fillable positions in the template by parsing the template and for generating code to extract and deposit fields from the machine instructions in source code into a precompiled template.

29. (Previously Presented) The dynamic binary translator system according to claim 26, wherein the dynamic binary translator arranged to be responsive to the machine instructions and wherein the dynamic binary translator outputs the translated target code instruction for processing on a target processor using the templates and fill and analysis routines during compilation of the dynamic binary translator.

30. (Previously Presented) A computer readable medium for performing a build and compilation process of a dynamic binary translator which operates between any processor or platform types and that translates a source code instruction into a target code instruction, having instructions that, when executed by a computer, cause the computer to perform a method comprising:
(a) creating a repository of magic numbers, wherein each magic number is a unique integer representing one and only one variant of a source instruction so that the collection of all the magic numbers represents the set of all valid instructions possible of a source instruction set to

form a set of constants associating a unique string with a unique integer, wherein the string shortly describes a variant of the instruction which the magic number is supposed to represent;

(b) creating a repository of templates, wherein each template is a functionally equivalent sequence of target architecture instructions, for each source instruction, each template having temporary native registers for its operations and is compiled at the compile time of the binary dynamic translator as a function and whose name is derived from the corresponding magic number's associated string so that the templates are available as a routine/function at run-time;

(c) creating a table, indexed on magic numbers, containing the references to the corresponding templates so that at run-time the translator can pick up the appropriate template for a given instruction after ascertaining its magic number;

(d) creating a repository of template filler routines in which each template filler routine has the knowledge of both the source and target instructions; causing the template filler routine of a given template to extract dynamic components of the source instruction and fill the extracted items in an in-memory template to replace corresponding place-holders; causing the translator to call the template filler routine for a given instruction at run-time so a template filler routine repository is linked to corresponding templates and magic numbers;

(e) creating a repository of minimal decode/magic-number-routines; causing a minimal decode/magic-number-extractor routines repository to give a raw source instruction and return the magic number of that instruction;

preparing the dynamic binary translator to operate between any processor or platform types by compiling steps (a)-(e) together.

31. (Previously Presented) The computer readable medium according to claim 30, wherein the dynamic binary translator maps the source code instructions to the target code instructions using the templates.

32. (Previously Presented) The computer readable medium according to claim 30, wherein the dynamic binary translator outputs the translated target code instruction for processing on a target processor using the templates and fill and analysis routines during compilation of the dynamic binary translator.

33. (New) Apparatus which operates between any platform types for translating machine instructions in source code into equivalent target instructions of a code of a target platform, wherein the source code differs from the code of the target platform, said apparatus comprising:

a processor; and

storage means coupled to the processor, wherein the storage means comprises:

a source of binary translation templates for mapping instructions in the source code into a set of instructions in the code of the target platform;

a fill and analysis routine generator arranged to be responsive to the templates for generating fill and analysis routines for identifying fillable positions in a template by parsing the template and for generating code to extract and deposit fields from the instructions in source code into a precompiled template; and

a dynamic binary translator of claim 21 arranged to be responsive to the precompiled template.